Using Reinforcement Learning for City Site Selection in the Turn-Based Strategy Game Civilization IV

Stefan Wender, Ian Watson

Abstract— **This paper describes the design and implementation of a reinforcement learner based on Q-Learning. This adaptive agent is applied to the city placement selection task in the commercial computer game *Civilization IV*. The city placement selection determines the founding sites for the cities in this turn-based empire building game from the *Civilization* series. Our aim is the creation of an adaptive machine learning approach for a task which is originally performed by a complex deterministic script. This machine learning approach results in a more challenging and dynamic computer AI. We present the preliminary findings on the performance of our reinforcement learning approach and we make a comparison between the performance of the adaptive agent and the original static game AI. Both the comparison and the performance measurements show encouraging results. Furthermore the behaviour and performance of the learning algorithm are elaborated and ways of extending our work are discussed.**

I. Introduction

One of the main incentives for integrating machine learning techniques into video games is the ability of those techniques to make those games more interesting in the long run through the creation of dynamic, human-like behaviour [1]. Among the most captivating games, especially in terms of long-term gameplay, are the games of the *Civilization* series. However these turn-based strategy games achieve their high replay value not through advanced adaptable AI techniques but through a high level of complexity in the later stages of the game. The early stages however are, as we will also see in this paper, mostly deterministic and therefore not very challenging. These characteristics as well as the large number of tasks involved in playing the game make *Civilization* games an ideal test bed where one of those many tasks can be replaced by a machine learning agent, thus making the AI less predictable and improving the overall game play experience.

II. Related Work

The machine learning method we chose for our task is Reinforcement Learning (RL) [2], a technique which allows us to create an adaptive agent that will learn unsupervised while playing the game. More specifically the Q-Learning algorithm as introduced by [3] will be used to demonstrate the applicability of reinforcement learning in the commercial video game *Civilization IV*.

Because of the broad spectrum of problems involved in Civilization video games as well as the multitude of versions of the games that are available, several of them with open source code, multiple variants of the game have been used in academic research. Perhaps most popular as a test bed is the Civilization variant *FreeCiv*, an open source version of the commercial game *Civilization II*. *FreeCiv* has been used to show the effectiveness of model-based reflection and self adaption [4]. Furthermore an agent for *FreeCiv* has been developed that plays the complete early expansion phase of the game [5].

The development of an AI module that is based on Case-Based Reasoning (CBR) for the open source Civilization clone *C-Evo* is documented in [6]. *C-Evo* is an open source variant of Civilization, which is closest related to *Civilization II* and allows for the development of different AI modules which can compete against each other.

More directly related to this paper is research which uses the commercial Civilization games as a test bed. The commercial Civilization game *Call To Power II* (CTP2) is used as a test bed for an adaptive game AI in [7]. In order to communicate with the game an ontology for the domain is developed and case-based planning in combination with CBR is used to create an adaptive AI. CTP2 has also been integrated with the test environment TIELT [8], thus preparing a test bed for future research using CTP2.

In [9] a Q-Learning algorithm is used to create an adaptive agent for the fighting game *Knock'em*. The agent is initially trained offline to be able to adapt quickly to the opponent in an online environment. RETALIATE (**Re**inforced **Ta**ctic **L**earning **i**n **A**gent-**T**am **E**nvironments), an online Q-Learning algorithm that creates strategies for teams of computer agents in the commercial First Person Shooter (FPS) game *Unreal Tournament* is introduced in [10]. This approach is extended in [11], where the authors use CBR in order to get the original RETALIATE algorithm to adapt more quickly to changes in the environment.

III. Civilization IV as Test Bed for Research in Computer Game AI

The variant of the Civilization game which will be used as a test bed in this paper is *Civilization IV*. *Civilization IV* is the latest title in the commercial series of the original game. Large parts of its code base, including the part which controls the AI, have been released as open source. Also the existing computer AI is already quite sophisticated and thus can provide a challenging opponent in empirical experiments. Furthermore an improvement of the existing AI would show that research from academia can be used to create a bigger challenge and thus offer a more enjoyable playing experience which will in the end lead to better games in general.

Stefan Wender and Ian Watson are with The University of Auckland, Department of Computer Science, Auckland, New Zealand; e-mail: swen011@aucklanduni.ac.nz || ian@cs.auckland.ac.nz

978-1-4244-2974-5/08/$25.00 ©2008 IEEE

However the use of *Civilization IV* as a test bed for research also bears a challenge. The code base that was released as open source consists of more than 100000 lines of code, which mostly are very sparingly commented. Since only parts of the source code have been released, several major functions have to be emulated, most importantly the automatic restarting of a game which is crucial when running tests that are supposed to last for several thousand games.

A. The Game

Civilization is the name of a series of turn-based strategy games. In these games the player has to lead a civilization of his choice from the beginnings BC to the present day. It involves building and managing cities and armies, advancing the own empire through research and expansion as well as interacting with other, computer-controlled civilizations through means of diplomacy or war in a turn-based environment. The popularity of the original game has lead to a multitude of incarnations of the game, both commercial and open source.

B. The City Placement Task

The most important asset in a game of Civilization IV are the cities. The three major resources a city produces are food (used for growth and upkeep of a city), commerce (used among others for research and income) and production (used to produce units and buildings). Furthermore special bonuses which grant additional basic resources or other benefits like accelerated building speed can be gained. The playing field in Civilization IV is partitioned into "plots" with each plot producing a certain amount of the resources mentioned above. A city can gain access only to the resources of a plot which is in a fix shape of 21 plots surrounding the city: Figure 1.

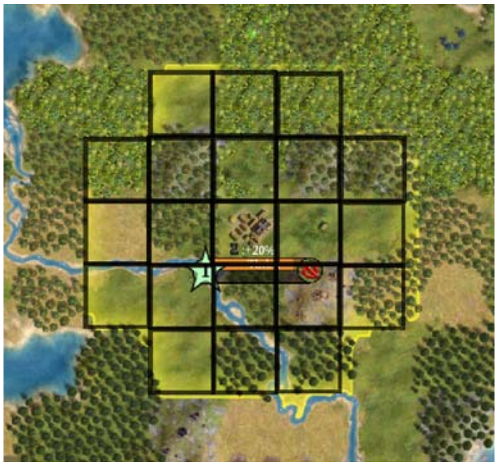


Fig. 1. Civilization IV: Workable City Radius

In addition the borders of an empire and thus the area of influence of its player are defined by the summarized borders of the cities of that empire. Therefore the placement of cities is a crucial decision and influences the outcome of a game to a large degree. This is the reason why we chose to apply RL to the city site selection task.

Cities are founded by mobile settler units which can be exchanged for a city on the plot they are located on. The standard AI uses a strictly sequential method of determining which are the best sites for building new cities. Each plot on the map is assigned a "founding value" which is based on numerous attributes like the position of the plot in relation to water, proximity of enemy settlements, proximity of friendly cities and of course the resources that can be gained from plots. We replace this sequential computation of founding values with reinforcement learning.

At the current stage of our research the task of the reinforcement learner is limited to choosing the best location for an existing settler. The decision of when to build a settler is still made by the original game AI.

IV. Reinforcement Learning Model

Reinforcement learning is an unsupervised machine learning technique, in which an agent tries to maximise the reward signal [2]. This agent tries to find an optimal policy, i.e. a mapping from states to the probabilities of taking possible actions in order to gain the maximum possible reward. The reinforcement learning model for the city placement task consisting of the set of states S, possible actions A and the scalar reward signal r is defined as follows:

A. States

A state $s \in S$ contains the coordinates of all existing cities of the active player. The other important information besides the position of a city is when this city was created. This information is crucial since a different order of founding can lead to very different results. Therefore, in order to satisfy the Markov property (i.e. any state is as well qualified to predict future states as a complete history of all past sensations up to this state would be) and thus for the defined environment to represent a Markov Decision Processes (MDP) each plot also contains the information when, in relation to the other cities, this city was founded.

A state $s \in S$ can be described as a set of triples *(X-Coordinate, Y-Coordinate, Rank in the Founding Sequence)* and each triple is representing one city. This definition of the states means that the resulting model will be a graph with no cycles, i.e. a tree. Figure 2 shows a part of such a tree with the nodes representing the states and branches representing the actions. Because of this structure of the state space, no state can be reached more than once in one episode.

The set of all states S consists therefore of all possible combinations of the *(X-Coordinate, Y-Coordinate, Rank in the Founding Sequence)* triples where cities can be built on any plot $p \in P$.The resulting size of the state space is

$$|S| = \sum_{i=0}^{c} \frac{|P|!}{(|P| - i)!} \tag{1}$$

With $c = |P|$ since every plot on the map could be a city.

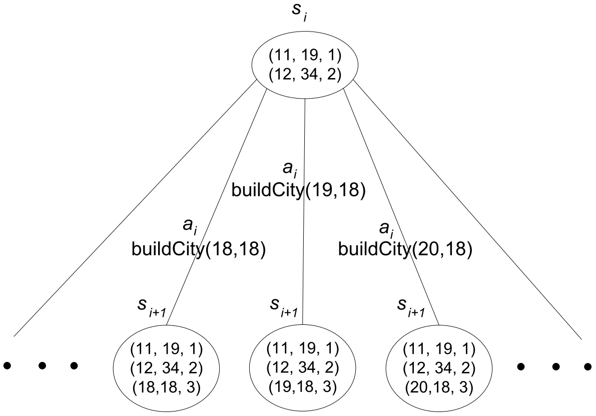


Fig. 2. Excerpt of the State Space S

B. Actions

The set A of possible actions which can be taken when in a state $s \in S$ consists of founding a city on any of the plots $(p \in P \| p \notin s)$, i.e. any plot where there is no city of the active player yet. Since the map size of a game varies and for an average sized map there are $|P| = 2560$ plots already, this results in a very large state space. One measure we took to reduce this size significantly is to ignore ocean plots, as cities can only be founded on land. This reduces the number of possible plots to about one third of the map size.

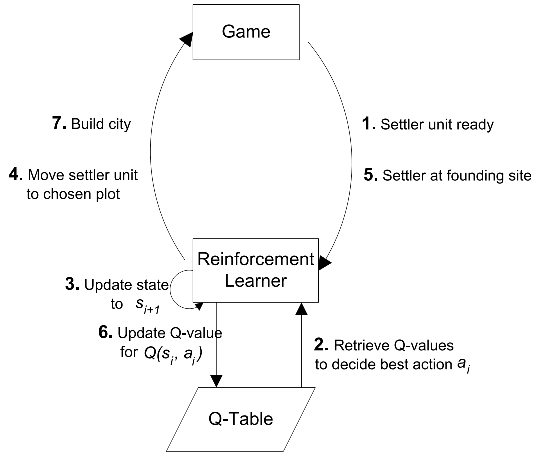
C. Rewards

The reward signal is based on the score of a player. In the original game this score is used to compare the performance of the players with each other and it is updated every turn for all players. The game score consists of points for population in the cities, territory (the cultural borders of the cities added up) and technological advancement (developed with research output from the cities). Therefore, all the parts the game score is made up of are connected to the cities. A time step, i.e. the time frame in which the reinforcement learner has to choose an action and receives a reward after performing that action, is defined as the time between founding one city and founding the next city. The update of the Q-value $Q(s_i, a_i)$ after taking an action a_i in state s_i happens immediately before executing the next action a_{i+1}, i.e. founding the next city. The selection of the appropriate plot for this foundation however can happen several game turns before that, with the settler unit moving to the chosen plot afterwards. The scalar value which represents the actual reward is computed by calculating the difference in game score between the founding turn of the last city and the founding turn of this city. The difference is then divided by the number of game turns that have passed between the two foundations:

$$r \leftarrow \frac{(GameScore_{new} - GameScore_{old})}{(GameTurns_{new} - GameTurns_{old})}$$

V. Algorithm

The top-level algorithm which controls the overall city site selection task can be seen in Figure 3.



Game
7. Build city
1. Settler unit ready
4. Move settler unit to chosen plot
5. Settler at founding site
Reinforcement Learner
3. Update state to s_{i+1}
6. Update Q-value for $Q(s_i, a_i)$
2. Retrieve Q-values to decide best action a_i
Q-Table

Fig. 3. Top-level Algorithm for City Site Selection

The actual Q-Learning algorithm which is based on *One-step Q-Learning* as described in [2] is shown below.

```
Initialise $Q(s,a)$ to 0
  Repeat for each episode:
    Initialise $s$
    Repeat for each step in this episode:
      Determine possible actions $A_i$ in $s_i$
      Choose action $a_i \in A_i$ as
        I) $a_i \leftarrow max\ Q(s_i, a_i)$ with probability 1-$\epsilon$
        OR
        II) $a_i \leftarrow random\ a_i \in A_i$ with probability $\epsilon$
      Send settler unit to plot chosen in $a_i$
      $r \leftarrow gameScoreGainPerTurn()$
      $Q(s_i, a_i)$
        $\leftarrow Q(s_i, a_i)$
          $+\alpha \left[r + \gamma * max_{a_{i+1}} Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i)\right]$
      $s_i \leftarrow s_{i+1}$
    until the number of steps is $X$
```

The setting of a fixed number of turns X is motivated by the game mechanics. The size of X is directly related to the size of the map and the number of opponents. The size is usually defined in a way that after X turns, most of the map has been divided between the different factions and the main focus in the game shifts from expansion to consolidation of acquired territory and conquest of enemy terrain and thus away from the city placement task.

The discount factor γ in our experiments, which are presented in section VI, is set to 0.1. The reason for this rather low value lies in the large number of possible actions in each state. All $Q(s, a)$ values are initialised to zero and the reward signal is always positive. Therefore, actions which are pursued in earlier episodes are likely to be called disproportionally more often if a policy different to complete randomness is pursued. The learning rate α is set to 0.2 which proved to be high enough to lead to relatively fast convergence while being low enough to protect the Q-values from anomalies in the reward signal. Both γ and α values have also been tested in trial runs and proven to work best for our purposes with the values given above.

Since Q-Learning is by definition an *off-policy* algorithm, the learned action-value function Q converges with probability one to the optimal action-value function Q^* even following a completely random policy. However, this requires visiting every single state infinite times, which is computationally not feasible. The algorithm uses an ϵ-greedy policy. Its general behavior can be modified by altering ϵ, depending on whether the main aim is learning or performing optimally, i.e. if the focus should be on exploration or exploitation.

It is noteworthy that all Q-values show a dependency on the starting position on the map of the respective player. Furthermore, since the usability of a plot as founding site for a city completely depends on the geography of the surroundings, the Q-values obviously are correlated to the map which is used in the game. As soon as the geography changes, the Q-values have to be computed from scratch. This dependency and how we intend to deal with it is further elaborated in section VII.

VI. Empirical Evaluation

We ran several experiments to compare our adaptive city plot selection algorithm to the existing deterministic sequential approach. In order to be able to compare the growing effectiveness of greater coverage over the state space, several experiments with differing numbers of turns as well as different settings for the ϵ-greedy policy were run. Except for the method of choosing the best spot for its cities, the original game AI was left unchanged.

The standard setup is one computer player that uses experience gained through Q-Learning against two other computer players which use the standard method of determining founding plots. The map used is the same in every game, as well as the starting positions. All players have exactly the same "character traits" (a game mechanic which leads to advantages in certain parts of the game like research or combat) so they are starting under exactly the same premises. The size of the chosen map is the second smallest in the game and the map consists of about 300 plots which can be settled. According to Equation (1) this leads to the number of possible states

$$|S| = \sum_{i=0}^{c} \frac{300!}{(300-i)!}.$$

c is in this case equal to the number of cities that are expected to be built in the given number of turns. This basically means that not the complete state-tree will be traversed but only the tree up to a depth equal to the maximum number of cities. The number of game turns differed between the experiments and was decided according to the goal of the respective test. Due to the low number of actions which are taken in one episode, a large number of episodes had to be played for every setup to get meaningful results.

One of our main aims was to find out how the reinforcement learner performed compared to the standard game AI. To achieve an adequate coverage of the state space, which is necessary to reach comparable results to the static but quite sophisticated standard game AI, the number of game turns was set to 50 (12.5 % of the maximum length of a game). This seems rather short but since the map on which the game is played is small, the game phase during which the players found new cities is usually very short as well. After 50 turns on average about half of the map has been occupied by the three players.

The limitation of the single episodes to a length of 50 turns leads to an expected maximum number of cities of 2 for the reinforcement player. This means that the possible states are limited to about

$$\sum_{i=0}^{2} \frac{300!}{(300-i)!} \approx 90000.$$

Since despite the low number of cities the high branching factor leads to this large state space, convergence is not guaranteed, even though the Q-Learning algorithm is usually able to cope through the ϵ-greedy policy which pursues the maximum rewards with probability $1-\epsilon$. ϵ was initialised at 0.9, i.e. in 90% of all cases our algorithm would pick a random action while selecting the action with the highest $Q(s,a)$ value in the remaining 10%. This ratio was subsequently slowly reverted, that means after 3000 played episodes only 10% of all actions would be random while 90% were picked according to the highest $Q(s,a)$ value. This is necessary to draw a meaningful comparison between the reinforcement learner and the standard AI when both try to play optimal or close to optimal.

3050 episodes of length 50 turns were played by the computer AI. After each episode the score of the players was recorded. For the final evaluation, the score of the RL player was averaged across the last 50 episodes to even out the anomalies which occur because of the explorative policy. As a reference value, the same experiment was performed with a standard AI player instead of the reinforcement player, i.e. three standard computer AI players compete against each other on the same map with the same premises as in the previous test. Figure 4 shows the results of both experiments.

The first thing that attracts attention is the performance of the standard computer AI under these circumstances, which results in the same score for every single game. This is due to the fact that there is no randomness in the decisions of the computer AI when it comes to city placement. There are very few non-deterministic decisions in *Civilization IV*, most of them in the combat algorithms, but those do not have any effect until later in the game when players fight against each other. Therefore, for the first 50 turns, the three standard AIs always performed the exact same actions.

The diagram also shows that the reinforcement learner,

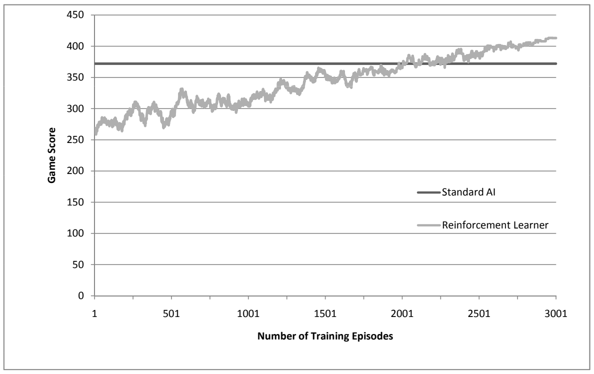


Fig. 4. Average Score Comparison between Reinforcement Learner and Standard AI for Games with Length 50 Turns

while initially inferior in average score, ends up beating the average score of the standard AI. On the downside it is noteworthy that it took the reinforcement learner more than 2000 episodes of the game to reach that point. Since the number of the episodes played is still a lot smaller than the state space for the placement of two cities, there also remains room for improvement by finding even better policies.

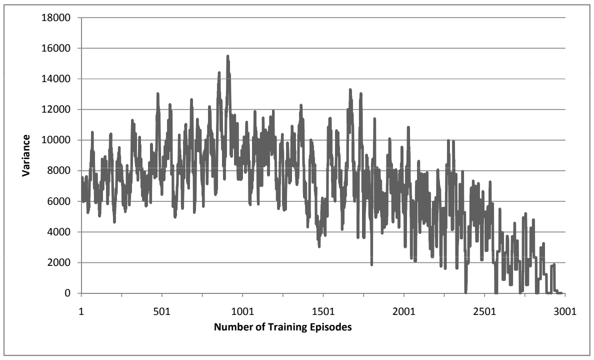


Fig. 5. Variance in the Average Score: From Explorative to Exploitative Policy by decreasing ϵ

Figure 5 shows the variance of the average of the scores for the reinforcement learner. Its decline illustrates the change of policy with increasing number of episodes.

Since the reinforcement learner performed very well for short games and a large number of episodes, another experiment was conducted to evaluate the convergence with less episodes and more turns. The number of turns was doubled to 100 turns per episode while the number of episodes was reduced to 1500. The evaluation happened in the same way as in the previous experiment through recording the game score and averaging the score for 50 episodes. As an addition the score was not only measured at the end of an episode, i.e. after 100 turns but also after 50 turns like in the previous experiment. Furthermore we performed another set of tests in the same environment with an AI that follows a completely random policy. This means that $\epsilon = 1$, which results in the player always picking actions at random.

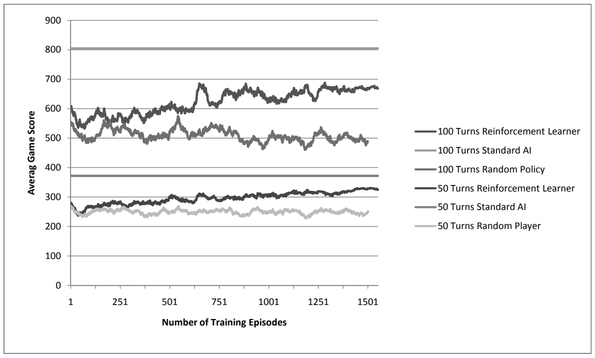


Fig. 6. Average Score Comparison for Games of Length 100 Turns

Figure 6 shows the results of these tests. As in the previous test run with only 50 turns, the standard AI achieves the same score for every single game at 50 turns. According to the diagram it also seems as if this is the case at 100 turns. However the standard AI achieves not the same score in every game, but alternates between two different scores which ultimately results in the same average over 50 episodes. This alternation suggests that a probabilistic decision is made during the second 50 turns. At the beginning when the RL agent has no experience yet, both RL and random agent have about the same average score. But while the score for the player following a completely random policy shows as expected no sign of long term growth or decline, the average score of the reinforcement learner improves with the growing number of episodes played.

The score for the reinforcement learner at 50 turns shows the same upward tendency as the score for the RL agent in the previous experiment (Figure 5). The average score is still lower than that of the standard AI because of the smaller number of episodes played. If growth of the average score continues, this would very likely change within the next 1000 episodes. The average score for the reinforcement learner after 100 turns shows the same tendency as the score at 50 turns even though the gap between the score. However the gap between the average score for the standard AI and the reinforcement learner is much bigger at 100 turns than at 50 turns. This can be explained through the significant difference in size of the state spaces for 50 turns and 100 turns. While during 50 turns players will get a maximum of two cities, 100 turns will allow building up to five cities which multiplies the number of possible states by nearly 300^3. Therefore it is remarkable that there is already a visible increase in the average score at 100 turns. This also means that there is potentially a huge margin to be gained over the standard AI through optimising the Q-values.

VII. Future Work

As previously stated, this paper presents the first results of a work in progress, the application of reinforcement learning to tasks in Civilization IV. Several extensions and additions are planned for the near future and can be divided into three categories. These categories are the improvement of the Q-Learning algorithm which has been used throughout this paper, the evaluation of other reinforcement learning algorithms and the combination of other machine learning techniques with RL. Furthermore the usage of *Civilization IV* as a test bed for computer game AI research can be extended.

The results of the empirical evaluation show that while the applied reinforcement learning method has potential to outperform the static standard AI, in its current state learning becomes computationally unfeasible when crossing a certain threshold of game complexity, either in matters of game turns or map plots. Therefore the optimisation of the Q-Learning algorithm is crucial to extend the tasks for which it can be used, i.e. longer games or larger maps. One way to to do this is by speeding up the learning process and as a result accelerating the convergence towards the optimal action-value function. This can for instance be achieved by using eligibility traces and thus having a multi-step Q-Learning algorithm instead of the current one-step Q-Learning. Another way to improve the speed of convergence is the initialisation. At the moment all $Q(s, a)$ values are initialised to 0 at the beginning of the algorithm. This means that every state has to be visited in order to determine its usefulness for the exploitation part of the policy, often only to conclude that its usefulness is very low. If the states were instead initialised to the precomputed ”founding values” that are used by the standard game AI, these values could serve as indicators about the usefulness of a plot for the reinforcement learner. This would not speed up the guaranteed convergence to an optimal policy π^* but generate better performance earlier on, resulting in more challenging gameplay.

Besides extending the existing method, other reinforcement learning algorithms and techniques such as the on-policy temporal-difference algorithm SARSA or Monte Carlo methods will be evaluated as to how well they are suited for the city placement selection task [2].

Another field for future research on RL using Civilization IV as a test bed is the combination of other machine learning methods with RL. One particularly promising method is Case-Based Reasoning (CBR). The application of CBR to the plot selection task would allow to resolve the previously mentioned problem with learned experience on one map being useless on another map because of the different topology.

Furthermore the application of Motivated Reinforcement Learning (MRL) could improve the game play experience. One of the game mechanics in Civilization IV are ”character traits” of the different computer AIs. Certain players have by definition advantages in certain areas of the game like expansion, finance or combat. These advantages are hard coded numbers and are meant to express certain character traits like aggressiveness or expansionism. If those agents would instead use a reinforcement learner which gets his rewards through a motivational function as described in [12], this could lead to very interesting behaviour for AI players.

Also the task for which these machine learning techniques are used can be extended. While at the moment the task only consists of determining where a city should be, in the future this could also include the choice if the city is needed at all, i.e. the optimal number of cities and when to build them.

VIII. Conclusions

This paper presents the design and implementation of a reinforcement learner which is used to perform a city site selection task in the turn-based strategy game *Civilization IV*. The Q-Learning algorithm which was used, manages to learn city placement strategies for specific maps. After sufficient training the RL agent outperforms the standard game AI in short matches. The reinforcement learner also shows promising results for longer and more complex games. The findings from these experiments on the possible shortcomings of the algorithm lay the groundwork for future work. Furthermore the usage of the commercial video game *Civilization IV* as a test bed for AI research demonstrates great potential because of the diversity of the tasks involved in Civilization and the relative ease with which these deterministic tasks can be taken over by machine learning agents. The integration of our RL agent into *Civilization IV* extends the otherwise deterministic early game by a non-deterministic, adaptable component which enhances the game-playing experience.

References

[1] J. Laird and M. van Lent, “Human-level AI’s Killer Application: Interactive Computer Games,” *AI Magazine*, vol. Summer 2001, pp. 1171–1178, 2001.

[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[3] C. Watkins, “Learning from Delayed Rewards,” Ph.D. dissertation, University of Cambridge, England, 1989.

[4] P. Ulam, A. Goel, and J. Jones, “Reflection in Action: Model-Based Self-Adaptation in Game Playing Agents,” in *Proceedings of the Nineteenth National Conference on Artificial Intelligence American Association for Artificial Intelligence (AAAI)*, 2004.

[5] P. A. Houk, “A Strategic Game Playing Agent for FreeCiv,” Northwestern University, Evanston, IL, Tech. Rep. NWU-CS-04-29, 2004.

[6] R. Sanchez-Pelegrin, M. A. Gomez-Martin, and B. Diaz-Agud, “A CBR Module for a Strategy Videogame,” in *1st Workshop on Computer Gaming and Simulation Environments, at 6th International Conference on Case-Based Reasoning (ICCBR)*, D. Aha and D. Wilson, Eds., 2005.

[7] A. Sanchez-Ruizy, S. Lee-Urban, H. Munoz-Avila, B. Diaz-Agudoy, and P. Gonzalez-Caleroy, “Game AI for a Turn-based Strategy Game with Plan Adaptation and Ontology-based Retrieval,” in *Proceedings of the ICAPS 2007 Workshop on Planning in Games*, 2007.

[8] D. W. Aha and M. Molineaux, “Integrating Learning in Interactive Gaming Simulators,” Intelligent Decision Aids Group; Navy Center for Applied Research in Artificial Intelligence, Tech. Rep., 2004.

[9] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, “Automatic computer game balancing: a reinforcement learning approach,” in *AAMAS ’05: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*. New York, NY, USA: ACM, 2005, pp. 1111–1112.

[10] M. Smith, S. Lee-Urban, and H. Muñoz-Avila, “RETALIATE: Learning Winning Policies in First-Person Shooter Games,” in *AAAI*, 2007.

[11] B. Auslander, S. Lee-Urban, C. Hogg, and H. Munoz-Avila, “Recognizing the Enemy: Combining Reinforcement Learning with Strategy Selection using Case-Based Reasoning,” in *Advances in Case-Based Reasoning: 9th European Conference, ECCBR 2008, Trier, Germany, September, 2008, Proceedings*, K.-D. Althoff, R. Bergmann, M. Minor, and A. Hanft, Eds. Springer, 2008.

[12] K. E. Merrick and M. L. Maher, “Motivated Reinforcement Learning for Adaptive Characters in Open-Ended Simulation Games,” in *ACE ’07: Proceedings of the International Conference on Advances in Computer Entertainment Technology*. New York, NY, USA: ACM, 2007, pp. 127–134.